What is claimed is:

1. An apparatus comprising:

marking logic to mark instruction information for an instruction of a speculative thread as speculative; and

blocker logic to prevent data associated with a store instruction of the speculative thread from being forwarded to an instruction of a non-speculative thread, the blocker logic further to prevent the data from being stored in a memory system.

2. The apparatus of claim 1, wherein:

blocker logic is further to allow the data associated with a store instruction of the speculative thread to be forwarded to an instruction of a second speculative thread.

3. The apparatus of claim 1, further comprising:

a plurality of store request buffers, each store request buffer including a speculation identifier field.

4. The apparatus of claim 1, wherein the memory system further comprises:

a data cache that includes a safe-store indicator field associated with each entry of a tag array.

5. The apparatus of claim 1, wherein:

the blocker logic is included within the memory system.

6. The apparatus of claim 1, wherein blocker logic further includes:

dependence blocker logic to prevent data associated with a speculative store instruction from being forwarded to an instruction of the non-speculative thread ; and

store blocker logic to prevent the data from being stored in a memory system.

7. The apparatus of claim 6, wherein:

store blocker logic is outside the execution pipeline.

8. The apparatus of claim 7, wherein:

store blocker logic is included in the memory system.

9. The apparatus of claim 6, wherein:

dependence blocker logic is included in an execution pipeline.

10. The apparatus of claim 9, wherein:

dependence blocker logic is included in a memory ordering buffer.

11. A system, comprising:

a memory system that includes a dynamic random access memory;

a processor including dependence blocker logic to prevent data associated with a store instruction of a speculative thread from being forwarded to an instruction of a non-speculative thread;

the processor further including store blocker logic to prevent the data from being stored in the memory system.

12. The system of claim 11, wherein:

the processor further includes marking logic to mark instruction information associated with the store instruction as speculative.

13. The system of claim 12, wherein:

the marking logic is further to associate a safe speculation domain ID with the instruction information.

14. The system of claim 13, wherein:

the marking logic is further to indicate a thread identifier as the speculation domain ID.

15. The system of claim 12, further comprising:

a store request buffer to store the speculation domain ID.

16. The system of claim 11, wherein:

the processor includes a first logical processor to execute the non-speculative thread; and

the processor includes a second logical processor to execute the speculative thread.

17. The system of claim 11, further comprising:

a second processor that includes said dependence blocker logic and said store blocker logic;

wherein said processor is to execute the non-speculative thread and said second processor is to execute the speculative thread.

18. The system of claim of claim 11, wherein:

the memory system includes a cache organized to include a plurality of tag lines, wherein each tag line of the cache includes a unique helper thread ID field.

19. The system of claim 11, wherein:

the memory system includes a cache organized to include a plurality of tag lines, wherein each tag line of the cache includes a safe-store indicator field.

20. The system of claim 11, wherein:

the memory system includes a victim tag cache to indicate evicted cache lines that include speculative load data.

21. A method, comprising:

receiving instruction information for a load instruction, the instruction information including a load address;

performing a dependence check, wherein performing the dependence check includes:

determining if a store address of an in-flight store instruction matches the load address; and

determining if the load instruction and the in-flight store instruction each originate with a speculative thread;

forwarding, if the dependence check is successful, store data associated with the in-flight store instruction to the load instruction; and

declining to forward, if the dependence check is not successful, the store data to the load instruction.

22. The method of claim 21, wherein performing the dependence check further comprises:

determining if the in-flight store instruction and the load instruction originate from the same thread.

23. The method of claim 22, wherein determining if the in-flight store instruction and the load instruction originate from the same thread further comprises:

determining if a thread ID associated with the in-flight store instruction matches a thread ID associated with the load instruction.

24. The method of claim 21, wherein performing the dependence check further comprises:

if the load instruction and the in-flight store instruction do not each originate with a speculative thread, determining if the load instruction and the in-flight store instruction each originate with a non-speculative thread.

25. The method of claim 21, further wherein:

declining to forward further comprises declining to forward the store data to the load instruction if (the load instruction and the in-flight store instruction each originate with a speculative thread) AND (the in-flight store instruction originates with a speculative thread that is not older in program order than the speculative thread from which the load instruction originates).

26. A method, comprising:

processing a speculative thread cache read request;

processing a speculative thread cache write request; and

processing a cache access request from a non-speculative thread.

27. The method of claim 26, wherein processing a speculative thread cache read request further comprises:

forwarding speculative data from a cache to a speculative thread responsive to a data cache read request.

28. The method of claim 26, wherein processing a speculative thread cache read request further comprises:

forwarding non-speculative store data from a cache to a speculative thread responsive to a data cache read request.

29. The method of claim 26, wherein processing a cache access request from a non-speculative thread further comprises:

forwarding non-speculative data from a cache to a non-speculative thread responsive to a data cache read request.

30. The method of claim 26, wherein processing a cache access request from a non-speculative thread further comprises:

if a cache does not include a cache line associated with the cache access request, allocating a new cache line;

wherein allocating a new cache line further comprises:

if the new cache line includes dirty speculative data, allocating the new cache line without generating a writeback operation; and

if the new cache line includes dirty non-speculative data, generating a writeback operation.

31. The method of claim 26, wherein processing a speculative thread cache write request further comprises:

allowing the speculative thread to write data to the cache if a cache line corresponding to the cache write request includes speculative data.

32. The method of claim 26, wherein processing a speculative thread cache write request further comprises:

if the cache line corresponding to the cache write request contains dirty non-speculative data in the cache line corresponding to the data cache request:

generating a writeback of the dirty non-speculative data;

allowing the speculative thread to write speculative data to the cache line; and marking the cache line as speculative.

33. The method of claim 26, wherein processing a speculative thread cache write request further comprises:

if the cache does not contain data in a cache line corresponding to the data cache address:

allocating a new cache line;

marking the new cache line as speculative; and

allowing the speculative thread to write speculative data to the new cache line.